**Project Design Phase-II**

**Data Flow Diagram & User Stories**

| Date | 31 January 2025 |
| --- | --- |
| Team ID | LTVIP2025TMID53108 |
| Project Name | FlightFinder |
| Maximum Marks | 4 Marks |

**Data Flow Diagrams:**

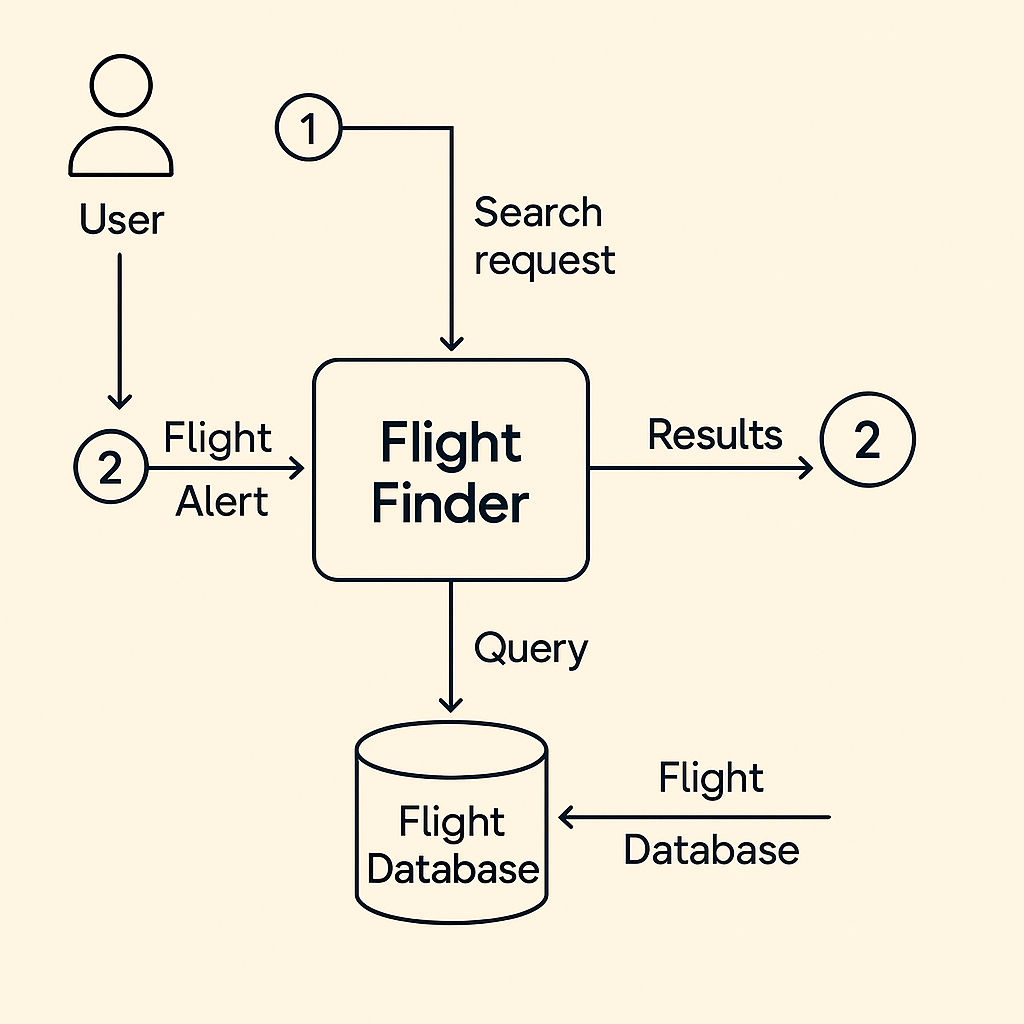
A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

Simplified:

**Main Process: Flight Search Engine (Process 0)**

This is the central processing unit that handles all incoming and outgoing data:

* Accepts **user search input**
* Sends queries to **Airline APIs**
* Processes results via filtering, indexing, and sorting
* Returns personalized results to the user
* Logs interaction metrics for admin analysis



**Context-Level (Level 0) DFD: FlightFinder**

**Purpose**: To provide a high-level overview of how external entities interact with the core *FlightFinder* system and how data flows among them.

📦 **External Entities**

* **User (Traveler)**: Initiates flight searches, sets preferences, receives results.
* **Airline APIs**: Provide flight schedules, real-time pricing, and seat availability.
* **Admin**: Manages system settings, monitors usage, analyzes performance metrics.

🔄 **Main Process: Flight Search Engine (Process 0)**

This is the central processing unit that handles all incoming and outgoing data:

* Accepts **user search input**
* Sends queries to **Airline APIs**
* Processes results via filtering, indexing, and sorting
* Returns personalized results to the user
* Logs interaction metrics for admin analysis

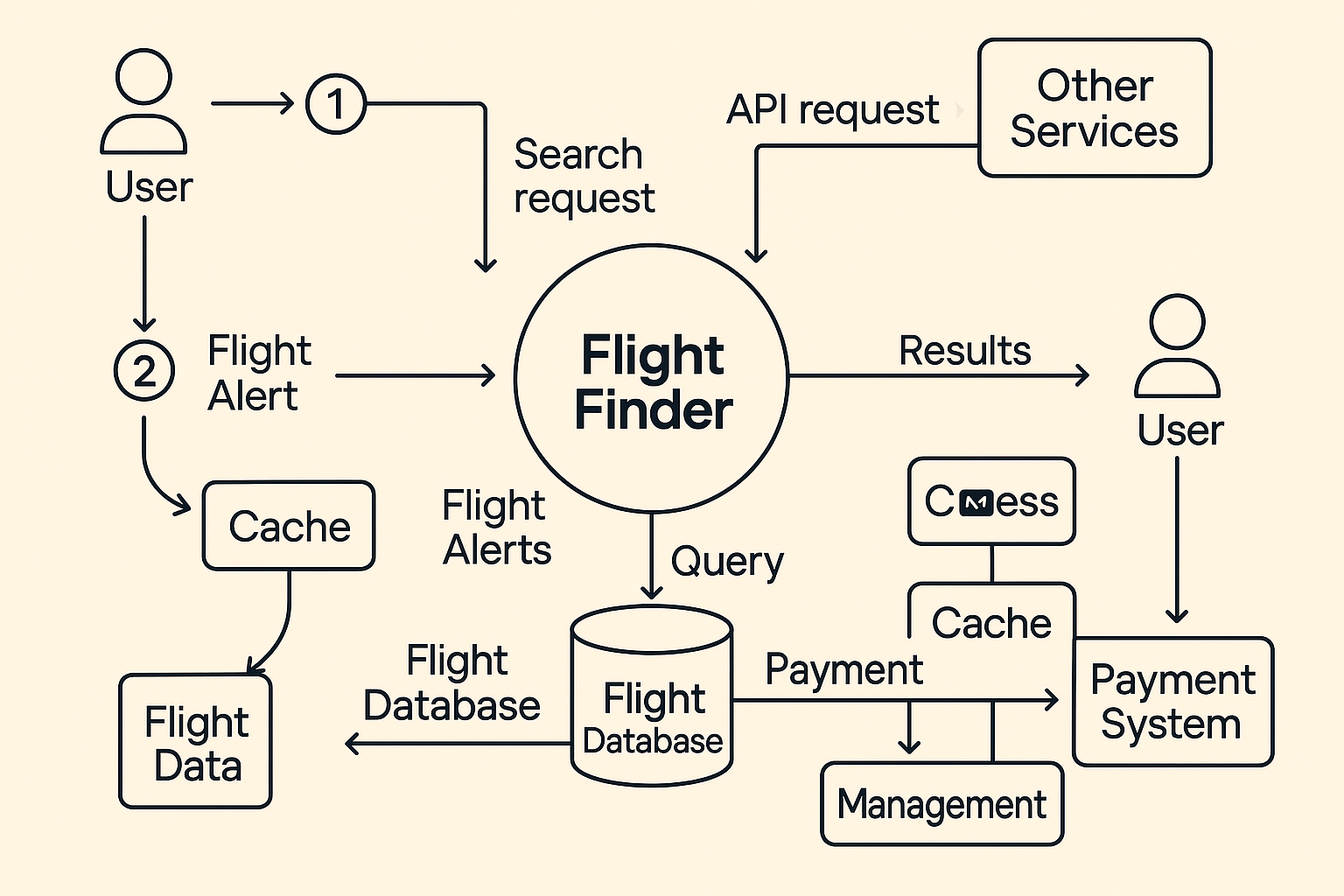
🗂️ **Data Stores**

* **Flight Database**: Stores cached results, indexed routes, and pricing history.
* **User Profiles**: Search history, saved alerts, preferences.
* **Analytics Store**: Logs performance metrics, popular queries, and error reports.

🔁 **Data Flows**

* User → Search Query → FlightFinder
* FlightFinder → API Calls → Airline APIs
* Airline APIs → Flight Results → FlightFinder
* FlightFinder → Filtered Results → User
* FlightFinder ↔ Flight Database: For caching & indexing
* Admin ↔ Analytics: For monitoring and optimization

If you’re ready, I can help sketch out a **Level 1 DFD** next—where each major subprocess (search, indexing, alerting) is broken down further. Want to dive into that?



Here's a concise table with key **user stories** for the *FlightFinder* project—focused and streamlined for quick reference:

**✈️ FlightFinder – Core User Stories (Mini Table)**

| **User Type** | **Epic** | **User Story** | **Priority** | **Sprint** |
| --- | --- | --- | --- | --- |
| Mobile User | Registration | Sign up with email and social logins | High | Sprint-1 |
| Mobile User | Search Flights | Search using filters (date, airline, etc.) | High | Sprint-2 |
| Mobile User | Alerts | Set price-drop alerts for favorite routes | High | Sprint-3 |
| Web User | Search & Book | Find and book flights via partner redirects | Medium | Sprint-2 |
| Customer Support | Ticket Handling | Respond to flight issues and user tickets | High | Sprint-3 |
| Administrator | Analytics Panel | Monitor user behavior and system performance | High | Sprint-2 |